Project Report

Project: Mini blockchain (simplified Bitcoin-like chain) implemented in Go

Report date: 2025-12-19

Course assignment goal: Build a simplified blockchain from scratch with: Block structure + hashing, Proof-of-Work (PoW), Persistence (DB), CLI interface, Wallets + UTXO transactions + signatures and 3-node P2P block synchronization

This report explains the **key architectural choices and optimizations** made across **Module 1 → Module 6** and documented the main **bugs/errors encountered** and how they were **diagnosed and resolved**.

Table of Contents

Executive Summary

Across Modules 1–6, the project makes a set of design choices that intentionally balance: (1) clarity for teaching, (2) correctness for a minimal UTXO system, (3) persistence and multi-node demonstration, and (4) small, testable modules.

Key architectural choices / unique designs:

1. **Progressive refactoring without rewriting modules**
   - Each module extends prior code (RAM → DB → CLI → UTXO → network) with minimal churn.
   - This reflected an upgradable and scalable interface.
2. **Block header uses a Merkle root rather than raw data**

- Once transactions arrive (Module 5), PoW commits to the Merkle root, instead of a free-form string.

3. **Versioned / per-node storage strategy**
   - We migrated from a single DB file to **per-node DB files** (`blockchain_<NODE_ID>.db`) to support multi-node simulation on one machine. This avoids shared state between nodes.
4. **Networking protocol is small but intentionally structured**
   - Messages are explicitly typed: `Version`, `GetBlocks`, `Inv`, `GetData`, `Block`.
   - Transport is TCP + gob, the protocol shape matches typical P2P blockchain sync patterns.
5. **Bitcoin-like RPC reads/writes on chain**
   - BoltDB locking makes multi-process DB access unreliable during demos. It will require a lot of stop/restart because of the locks on DB
   - We make the node processes owns their DB and the CLI queries state via network requests. This ensures no constant stop/restart and removes DB lock timeouts for read commands.
6. **Security model**
   - Transactions are signed and verified with ECDSA.
   - Address format uses Base58Check-like checksum.
   - The implementation prioritizes understanding and clean modularity over hardened threat resistance.

This section documents a repeatable demo flow

0) Enter project directory

```
Set-Location "C:\YOUR_PATH\my-blockchain"
```

1) Create wallets

```
PS C:\Users\15562\文档\FDU\Blockchain\my-blockchain> go run . createwallet
>> go run . createwallet
>> go run . listaddresses
New address: 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw
New address: 1Dn8HDYA5594HMKxTLTJkYj94WTpbeWs8B
```

2) Initialize blockchain on node 3000

```
PS C:\Users\15562\文档\FDU\Blockchain\my-blockchain> $env:NODE_ID='3000'
>> go run . createblockchain -address 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw
>> go run . getbalance -address 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw
Done! Created a new blockchain.
Balance of '18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw': 10
```

3) Start 3 nodes (3 terminals)

Terminal 1:

```
PS C:\Users\15562\文档\FDU> Set-Location "C:\Users\15562\文档\FDU\Blockchain\my-blockchain"
>> $env:NODE_ID='3000'
>> go run . startnode -miner 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw
2025/12/29 17:50:34 Node localhost:3000 listening (db=blockchain_3000.db, miner=18DFYQAyNnZKrQ6mj
vj2jDGcCYSehg1Waw)
```

Terminal 2:

```
PS C:\Users\15562\文档\FDU> Set-Location "C:\Users\15562\文档\FDU\Blockchain\my-blockchain"
>> $env:NODE_ID='3001'
>> go run . startnode
2025/12/29 17:50:44 Node localhost:3001 listening (db=blockchain_3001.db)
```

Terminal 3:

```
PS C:\Users\15562\文档\FDU> Set-Location "C:\Users\15562\文档\FDU\Blockchain\my-blockchain"
>> $env:NODE_ID='3002'
>> go run . startnode
2025/12/29 17:50:53 Node localhost:3002 listening (db=blockchain_3002.db)
```

4) Send a transaction (mines a new block) and verify balances and chain replication

```
PS C:\Users\15562\文档\FDU> Set-Location "C:\Users\15562\文档\FDU\Blockchain\my-blockchain"
>> $env:NODE_ID='3000'
>> go run . send -from 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw -to 1Dn8HDYA5594HMKxTLTJkYj94WTpbeWs8B
-amount 4
Success! Transaction accepted and mined into a new block by node.
PS C:\Users\15562\文档\FDU\Blockchain\my-blockchain> $env:NODE_ID='3000'; go run . getbalance -ad
dress 18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw
>> $env:NODE_ID='3000'; go run . getbalance -address 1Dn8HDYA5594HMKxTLTJkYj94WTpbeWs8B
Balance of '18DFYQAyNnZKrQ6mjvj2jDGcCYSehg1Waw': 16
Balance of '1Dn8HDYA5594HMKxTLTJkYj94WTpbeWs8B': 4
PS C:\Users\15562\文档\FDU\Blockchain\my-blockchain> $env:NODE_ID='3001'; go run . printchain
>> $env:NODE_ID='3002'; go run . printchain
===== Block 0 =====
Timestamp: 1767001888
Prev. hash: 0000760cff2c477da292095a8481ee169c6b0f10d42e186615e4075c46a8d6fe
Hash: 0000b7f50af14fbe1eb57defdcab8af5b4f058936164477c4432fd311e2775f8
Nonce: 130937
Merkle: dce70bd1d9b70140207e9c363daba47d2a8a08808ffc59d72efed7513c53be65
Tx count: 2
  TxID: c913de0826d213f84b78519cded607c95de1de3a33f4741bb630d0a35e81f933
  TxID: 4cdfc52cbbabf2d58e23b5c49722d16ffd8dde86a64e342f5b86074fe330668e

===== Block 1 =====
Timestamp: 0
Prev. hash:
Hash: 0000760cff2c477da292095a8481ee169c6b0f10d42e186615e4075c46a8d6fe
Nonce: 75472
Merkle: 3a9816aae1147ebeae58ae96250b4c129e8fcf43f88d9d64b56adaf68ba410d9
Tx count: 1
  TxID: f10af815b8a93f2894c62b2264657dd37882f3fe7d5c98b1c4f3590940bfd033

===== Block 0 =====
Timestamp: 1767001888
Prev. hash: 0000760cff2c477da292095a8481ee169c6b0f10d42e186615e4075c46a8d6fe
Hash: 0000b7f50af14fbe1eb57defdcab8af5b4f058936164477c4432fd311e2775f8
Nonce: 130937
Merkle: dce70bd1d9b70140207e9c363daba47d2a8a08808ffc59d72efed7513c53be65
Tx count: 2
  TxID: c913de0826d213f84b78519cded607c95de1de3a33f4741bb630d0a35e81f933
  TxID: 4cdfc52cbbabf2d58e23b5c49722d16ffd8dde86a64e342f5b86074fe330668e

===== Block 1 =====
Timestamp: 0
Prev. hash:
Hash: 0000760cff2c477da292095a8481ee169c6b0f10d42e186615e4075c46a8d6fe
Nonce: 75472
Merkle: 3a9816aae1147ebeae58ae96250b4c129e8fcf43f88d9d64b56adaf68ba410d9
Tx count: 1
  TxID: f10af815b8a93f2894c62b2264657dd37882f3fe7d5c98b1c4f3590940bfd033
```

Conclusion